Nek5000 Developers' Guide

1 Development of Single Element Operators, 1D

This section introduces practical considerations in the development and implementation of the spectral element methods in Nek5000. A brief overview of basic theory is provided for requisite background and notation. Code examples are provided in f77.

1.1 Theoretical Background: 1D

We begin with a brief introduction of typical spectral element (SE) operators in one space dimension, using just one spectral element on the interval $x \in \Omega := [-1, 1]$.

Consider the 1D Poisson problem, $-\nabla^2 u = f$, $u(-1) = u(1) = 0$. The weighted residual formulation for this problem amounts to finding an approximation $u(x) = \sum_j \hat{u}_j \phi_j(x)$, where the $\hat{u}_j$s are the unknown basis coefficients and $\{\phi_j(x)\}$, $j = 1, \ldots, n$ forms a linearly independent set of functions spanning a chosen finite dimensional subspace. To simplify matters, we presently restrict our attention to subspaces χ_0^N that automatically satisfy the homogeneous boundary conditions (hence the subscript 0)—this restriction will be lifted when we later consider DG-based implementations. In this particular (1D, single element) case we take $\chi^N \equiv \mathbb{P}_N(x)$, the space of all polynomials of degree $\leq N$, and χ_0^N to be the set of functions in χ^N that satisfy the boundary conditions. A simple counting argument shows that $n = N - 1$. Because it is relatively easy to implement Dirichlet or Neumann conditions in the context of iterative solvers, we will defer details of boundary conditions until later.

Thus, with some abuse of notation, we reformulate the Poisson problem into the variational problem, *Find* $u \in \chi_0^N$ *such that*

$$\int_\Omega \frac{dv}{dx}\frac{du}{dx}\,dx \;=\; \int_\Omega v\, f\, dx \qquad \forall\, v \in \chi_0^N. \tag{1}$$

Equation (1) constitutes the fully discretized 1D Poisson problem using a Galerkin formulation. Because v and u come from the same space, we have as many conditions (equations) to be satisfied as unknowns.

To make things concrete, we consider a nodal formulation with

$$\begin{aligned} u(x) &= \sum_{j=0}^{N} u_j h_j(x), \\ v(x) &= \sum_{i=0}^{N} v_i h_i(x), \end{aligned} \tag{2}$$

where $h_j(x)$ is the Lagrange interpolating polynomial satisfying $h_j(x_i) = \delta_{ij}$ and δ_{ij} is the Kronecker delta function. With this choice, one has $u(x_i) \equiv u_i$, such that the basis coefficients represent nodal values. (We therefore drop the hats on the coefficients.) Inserting the expansions (2) into (1) yields:

Find u_j such that $u \in \chi_0^N$ and

$$\sum_{i=0}^{N}\sum_{j=0}^{N} v_i \left(\int_{-1}^{1} \frac{dh_i}{dx}\frac{dh_j}{dx}\, dx \right) u_j \;=\; \sum_{i=0}^{N}\sum_{j=0}^{N} v_i \left(\int_{-1}^{1} h_i(x)h_j(x)\, dx \right) f_j \tag{3}$$

$\forall\, v_i$ such that $v \in \chi_0^N$. (Note that we have replaced $f(x)$ by its nodal interpolant. This substitution is valid because the error incurred is of the same order as the truncation error associated with seeking u in χ^N.) Letting $\underline{u} := (u_1, \ldots, u_n)^T$ and $\underline{v} := (v_1, \ldots, v_n)^T$ and assuming $-1 = x_0 < x_1 < \ldots < x_N = 1$, we can express (3) in matrix form, *Find $\underline{u} \in \mathbb{R}^n$ such that*

$$\underline{v}^T A \underline{u} = \underline{v}^T B \underline{f} \qquad \forall\, \underline{v} \in \mathbb{R}^n, \tag{4}$$

with

$$A_{ij} \;:=\; \int_{-1}^{1} \frac{dh_i}{dx}\frac{dh_j}{dx}\, dx, \tag{5}$$

$$B_{ij} \;:=\; \int_{-1}^{1} h_i(x)h_j(x)\, dx. \tag{6}$$

Because (4) must hold for all $\underline{v} \in \mathbb{R}^n$, we can write simply $A\underline{u} = B\underline{f}$.

Note that, aside from the boundary treatment (yet to be discussed), (5) could just as easily be in modal form if one took $h_i(x)$ to be something other than a Lagrange interpolant. Stability restricts our attention to certain acceptable classes of basis functions $h_i(x)$. If modal, they are typically linear combinations of orthogonal polynomials. If nodal, the nodes are typically chosen to be the zeros of orthogonal polynomials or, equivalently, quadrature points associated with a particular Gauss quadrature rule—such distributions tend to cluster points toward the ends of the interval. Virtually all such combinations ensure stability, that is, reasonably small condition numbers of the resultant matrices. One class of polynomials that is notoriously bad is the set $h_i = x^i$, which yields condition numbers that grow as $O(N!)$. Equally undesirable are Lagrange interpolants based on uniform point distributions.

In the SEM, one typically (though not always) employs numerical quadrature to evaluate A and B. If we take $x_i = \xi_i$, with $\xi_i \in [-1, 1]$ the Gauss-Lobatto-Legendre (GLL) quadrature points (Lobatto implies that the endpoints are included) and associated weights ρ_i, then any polynomial integrand of order up to $2N - 1$ will be evaluated exactly. In the SEM, one thus uses:

$$A_{ij} \;\equiv\; \sum_{k=0}^{N} \rho_k \left.\frac{dh_i}{dx}\right|_{\xi_k} \left.\frac{dh_j}{dx}\right|_{\xi_k}, \tag{7}$$

$$B_{ij} \;\approx\; \sum_{k=0}^{N} \rho_k h_i(\xi_k)h_j(\xi_k) \;=\; \rho_i \delta_{ij}. \tag{8}$$

Thus a diagonal mass matrix, B, results from the GLL quadrature. The error incurred is again on par with the truncation error. In this constant-coefficient 1D case, no error is incurred in using quadrature for the stiffness matrix A.

Boundary Conditions

As mentioned earlier, we've neglected detailed boundary condition treatment to simplify the introduction of A and B. For basis sets satisfying $h_0(x_i) = \delta_{i0}$ and $h_N(x_i) = \delta_{iN}$ (a desirable property

in modal bases and automatically satisfied with nodal bases), one can simply restrict the range of the indices i and j. (There is no issue with the integration index k—it always ranges from 0 to M, where M is the number of points associated with the chosen quadrature rule. We will later encounter cases where $M > N$, with quadrature weights and points denoted as ρ_k^M and ξ_k^M.) For the case of homogeneous Dirichlet conditions on each end of the interval, we take $i, j \in \{1, \ldots, N-1\}^2$. We occasionally denote the larger matrices having $i, j \in \{0, \ldots, N\}^2$ as $\bar{A}$ and $\bar{B}$, particularly when considering inhomogeneous boundary conditions, but typically drop the bars for simplicity (See [**?**] for more detail regarding boundary conditions.)

Special Notation for 1D, Single Element

For the very special 1D case where $\Omega = \hat{\Omega} := [-1, 1]$ we define $\hat{B} := \rho_i \delta_{ij}$ and the 1D derivative matrix,

$$\hat{D}_{ij} \;:=\; \left.\frac{dh_j}{dx}\right|_{\xi_i} \qquad i, j \in \{0, \ldots, N\}^2. \tag{9}$$

From these building blocks on obtains a particularly simple form for the stiffness matrix, $\hat{A} := \hat{D}^T \hat{B} \hat{D}$, which is clearly symmetric semi-positive definite. $\hat{A}$ has a one-dimensional null space associated with the constant vector $(1, 1, \ldots, 1)^T$ because $\hat{D}$ is rank-deficient. (Why?) These "hat" matrices are central building blocks in the multidimensional multi-element SE formulation.

1.2 1D Implementation

The f77 routine below will solve the 1D Poisson problem on a single element.

```
c-----------------------------------------------------------------------
      program sem1d  ! Single element SEM Poisson solver

      parameter (nmx=40,nm2=nmx*nmx)
      real ah(nm2),bh(nmx),ch(nm2),dh(nm2),zh(nmx),w(nm2)

      real u(nmx),f(nmx),ue(nmx)
      real a(nm2)

      write(6,*) 'Input polynomial degree, N:'
      read (5,*) n

      nr = n+1             ! number of points in r direction
      if (nr.gt.nmx) then
         write(6,*) 'n must be less than',nmx-1,'. Decrease n:',n
         call exitt
      endif

      ntot = nr           ! total number of points

      call semhat   (ah,bh,ch,dh,zh,n)    ! Generate 1D matrices: Ah,Bh, etc.

      call make_rhs (f,ue,z,ntot)         ! construct rhs and uexact
      call col2     (f,bh,ntot)           ! f <- B*f  (1D only, because B=bh)
```

```
                                                    _ T
      call get_mat_interior(a,ah,nr)      ! restrict A = RAR   for bcs (DFM02)
      n1 = nr-2                           ! number of points after restriction

      call gaujord       (a,n1,n1,w,ierr) ! Invert A.  Sloppy, but effective.

                                          !                          _
      u(1 ) = 0.                          ! set endpoint values for u
      u(nr) = 0.
                                          !     -1 _                    __
      call mxm(a,n1,f(2),n1,u(2),1)       ! u = A  B f ;  _interior_ of Bf only

      call err_chk(u,ue,ntot,n)

      call exitt                          ! custom exit handler (for MPI, etc.)
      end
c-----------------------------------------------------------------------
```

The required routines are provided in the Nek5000 svn repository.

```
c-----------------------------------------------------------------------
      subroutine make_rhs (f,ue,x,n)    ! construct rhs and uexact
      real f(1),ue(1),x(1)

      one   = 1.
      pi    = 4.*atan(one)

      k = 2
      do i=1,n
         argx  = k*pi*x(i)           ! Assume x \in [-1,1]
         ue(i) = sin(argx)           ! be certain that ue vanishes on dOmega !
         f (i) = k*k*pi*pi*ue(i)
      enddo

      return
      end
c-----------------------------------------------------------------------
      subroutine err_chk(u,ue,ntot,n)
      real u(1),ue(1)

      umax = glamax(u,ntot)  ! max of |u|, over all processors

      emax = 0.
      do i=1,ntot
         err  = abs(u(i)-ue(i))
         emax = max(emax,err)
      enddo
      emax = glmax(emax,1)  ! max of emax over all processors

      emax = emax / umax    ! Normalize

      write(6,1) n,emax
    1 format(i6,1pe14.4,' error')

      return
      end
c-----------------------------------------------------------------------
      subroutine get_mat_interior(ai,a,n)  ! restrict A to interior elements
      real ai(2:n-1,2:n-1),a(n,n)
```

```
      do j=2,n-1
      do i=2,n-1
         ai(i,j) = a(i,j)
      enddo
      enddo

      return
      end
c-----------------------------------------------------------------------
```

2 Development of Single Element Operators, 2D

We extend the 1D development of Section **??** to the two dimensional case making extensive use of the tensor-product forms that are central to the efficiency of the spectral element method in higher space dimensions.

Proceeding as in the 1D case, we again consider the Poisson model problem in $\Omega \subset \mathbb{R}^2$. For simplicity, we consider homogeneous Dirichlet conditions on $\partial\Omega$. The variational formulation for this problem reads, *Find* $u \in X_0^N$ *such that*

$$(\nabla v, \nabla u) = (v, f) \qquad \text{for } v \in X_0^N, \tag{10}$$

with the L^2 inner product $(f, g) := \int_\Omega f g \, dV$.

Initally, we consider the single element case on the square, $\Omega \equiv \hat{\Omega} := [-1, 1]^d$, with $d = 2$ and use a tensor product of our 1D polynomial basis functions as the basis for X^N. Thus, any function $u(r, s) \in X^N$ can be written as

$$u(r, s) = \sum_{j=0}^{N} \sum_{i=0}^{N} u_{ij} \, h_i(r) h_j(s), \tag{11}$$

where the h_is are the 1D Lagrange interpolants of the previous section. Note that, as in the 1D case, $u(\xi_p, \xi_q) \equiv u_{pq}$, which implies that our basis coefficients are also gridpoint values. Numerical solution of the model problem reduces to finding the basis coefficients, u_{ij}, $i, j \in \{0, \ldots, N\}^2$. Because of the homogeneous boundary conditions, we automatically have $u_{0:} = u_{N:} = u_{:0} = u_{:N} = 0$, which of course is equivalent to restricting the sums in (11) to range over $\{1, \ldots, N-1\}^2$.

On $\hat{\Omega}$, the variational statement (10) becomes:

$$\int_{-1}^{1}\int_{-1}^{1} \frac{\partial v}{\partial r}\frac{\partial u}{\partial r} + \frac{\partial v}{\partial s}\frac{\partial u}{\partial s} \, dr \, ds \;=\; \int_{-1}^{1}\int_{-1}^{1} v \, f \, dr \, ds \qquad \forall v \in X_0^N. \tag{12}$$

Inserting expansions of the form (11) into the above, we have

$$\sum_{ij,pq} v_{ij} \left[\int_{-1}^{1}\int_{-1}^{1} \left(\frac{dh_i}{dr} h_j(s)\right)\left(\frac{dh_p}{dr} h_q(s)\right) + \left(h_i(r)\frac{dh_j}{ds}\right)\left(h_p(r)\frac{dh_q}{ds}\right) \, dr \, ds \right] u_{pq} \;=\; \sum_{ij,pq} v_{ij} \left[\int_{-1}^{1}\int_{-1}^{1} \left(h_i(r)h_j(s)\right)\left(h_p(r)\right.\right.$$

Rearranging terms, we have

$$\sum_{ij,pq} v_{ij} \left[\left(\int_{-1}^{1} h_j(s) h_q(s) \, ds\right)\left(\int_{-1}^{1} \frac{dh_i}{dr}\frac{dh_p}{dr} \, dr\right) + \left(\int_{-1}^{1} \frac{dh_j}{ds}\frac{dh_q}{ds} \, ds\right)\left(\int_{-1}^{1} h_i(r) h_p(r) \, dr\right) \right] u_{pq} \;=\; \sum_{ij,pq} v_{ij} \left(\int_{-1}^{1} h_j(r) h_q(\right.$$

which clearly reveals the role of the 1D operators of the previous section. Once again we replace all 1D integrals with numerical quadrature and the discrete variation statement takes the form

$$\sum_{ij,pq} v_{ij} \left(\hat{B}_{jq}\hat{A}_{ip} + \hat{A}_{jq}\hat{B}_{ip} \right) u_{pq} \;=\; \sum_{ij,pq} v_{ij} \hat{B}_{jq}\hat{B}_{ip} f_{pq}. \tag{15}$$

Let $\underline{u} = \{u_{ij}\}$, $i, j \in 1, \ldots, N-1^2$ with similar expressions for $\underline{v}$ and $\underline{f}$. Then 16 can be written in compact form as, *Find* $\underline{u} \in \mathbb{R}^n$ *such that*

$$\underline{v}^T(\hat{B} \otimes \hat{A} + \hat{A} \otimes \hat{B})\underline{u} = \underline{v}^T(\hat{B} \otimes \hat{B})\underline{f} \qquad \forall \underline{v} \in \mathbb{R}^n, \tag{16}$$

or simply $A\underline{u} = B\underline{f}$, with $A := \hat{B} \otimes \hat{A} + \hat{A} \otimes \hat{B}$ defining the stiffness matrix and $A := \hat{B} \otimes \hat{B}$ the mass matrix.